The Temporal Dimension in End User Web Programming

Eytan Adar
University of Washington, CSE
101 Paul G. Allen Center
Seattle, WA 98195
eadar@cs.washington.edu

James Fogarty
University of Washington, CSE
101 Paul G. Allen Center
Seattle, WA 98195
jfogarty@cs.washington.edu

Mira Dontcheva
Advanced Technology Labs
Adobe Systems
San Francisco, CA 94103
mirad@adobe.com

Daniel S. Weld
University of Washington, CSE
101 Paul G. Allen Center
Seattle, WA 98195
weld@cs.washington.edu

CHI 2009....

Abstract
Despite the dynamic nature of the Web, most people view a static snapshot. Search engines, browsers, and higher level end-user programming environments only support observing and manipulating a single point in time—the "now." We propose that moving beyond this static viewpoint is important because a) maintaining a temporal view of the Web allows users to more clearly understand the behavior of their "programs" both in static and dynamic contexts and b) temporally changing information on the Web is interesting in its own right. In this paper we discuss the opportunities and challenges of integrating the temporal dimension in end-user programming environments and our experiences with *Zoetrope*, a tool for interacting with the ephemeral (i.e. dynamic) Web.

Introduction
Despite its dynamic nature, most interaction with the Web is primarily through a static snapshot—the *Now Web*. The *Now Web* is the view experienced through browsers, search engines, and most other end-user tools. This static version, as depicted in Figure 1, reflects the "current" version of all pages (e.g., CNN's current home page, the current traffic conditions between Seattle and Redmond, or the current bid price for a particular item on E-bay).

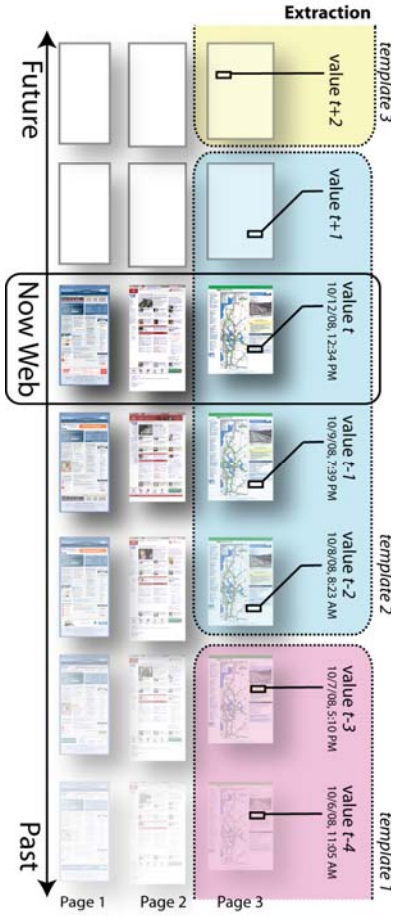

Figure 1. A view of the Now Web in the context of the Web's past and future.

By concentrating on the current snapshot, our Web tools and services largely ignore, or worse, discard historical versions of pages, thereby losing the temporal data itself (e.g., *historical* Amazon sales rank, traffic conditions, and webcam data). Thus, any historical versions of the Web become *ephemeral* as data irretrievably vanishes from the Web. For example, in Figure 1, any extraction before time *t* (the *Now*) on Web page 3 will be impossible unless the user had the foresight to crawl and archive the information.

As importantly, by failing to maintain past versions of pages, systems squander valuable training and testing data for end user programs and applications. By testing on the historical Web, programs intended to function both in the *Now* and in the future can be improved to identify and address possible failures. In Figure 1, for example, we note three different templates for page 3. Crafting an extraction or program solely on the current template (2) would result in a failure when the page switches to a new template.

In this paper we propose that the maintenance of historical Web versions can provide a powerful resource for engineering and improving the robustness of end-user programs on the *Now Web*. Zoetrope [1], a first prototype in this space, is a tool for interacting with the ephemeral Web. Though originally intended to address the issue of lost temporal data, the design of Zoetrope also illustrates the use of historical information to provide increased robustness in user programs and queries. We believe that the interaction techniques, widgets, and architecture of Zoetrope have implications for a broader set of tools and end-user programming environments. However, the use of this historical Web requires answering a number of research questions.

Although the historical Web represents a tremendous, untapped resource, breaking away from the familiar *Now* introduces an equally great challenge in leveraging historical information in an effective and intuitive way.

Zoetrope background

When someone is interested in finding how a particular piece of Web information has changed over time, they hope that someone has collected the data and made it searchable and available. While such historical data might be hard to find, it is likely easier to specify. We believe that a user will likely more rapidly identify the *present* version of that information. Thus, we designed Zoetrope that allows users to specify the data of interest from *within the context* of the *Now* version. Zoetrope supports this type of interaction and allows users to visually select any information on a webpage though a widget we call a *lens* (see Figure 2). Abstractly, a lens defines a query—specified visually as a rectangular widget—over the history of the page, which is maintained in a specialized index. Extracting the data contained in the lens, at every time step, Zoetrope reconstructs (and visualizes) a time series.

This functionality is implemented with a rapid crawling system (i.e. hourly downloads) that captures the "true" state of the page (saving all included images and content and freezing the DOM structure of the document) by running a modified Firefox browser in a server configuration. Through this dense sequence of historical page snapshots, users can reconstitute a fine-grained time series for the data of interest. By maintaining historical records of pages and data, end-user tools, such as Zoetrope, can better support information needs that extend beyond the *Now Web*.

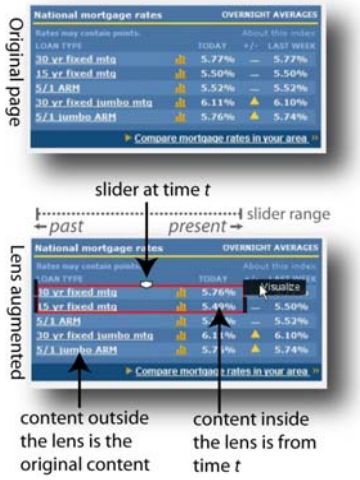

Figure 2. The anatomy of a lens

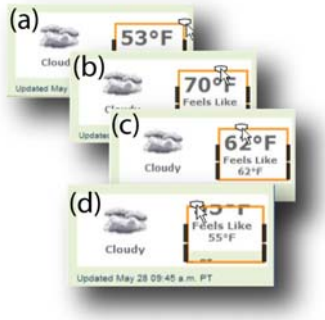

Figure 3. The detection of a failed extraction of the temperature (d)

Learning from the Past

A key feature of Zoetrope is the interactive specification of information of interest. Users can simply draw a lens anywhere on a webpage and track the data throughout the history of that page. Depending on how the information moves (or doesn't), different lenses support the tracking of visually stable, structurally stable, or content stable data. Visually stable information appears in the same location on the rendered page. Structurally stable information appears in the same place in the DOM tree. Finally, content-stable information may move on the page or within the DOM but does not vary significantly in terms of content (e.g., the Boston Red Sox's ranking in a list of teams may vary over the season, but the row corresponding to this information will always say "Boston Red Sox").

Example Generation

A key component of the Zoetrope lens is a slider at the top of every selection that allows a user to instantly flip the selection to a different time. This interaction technique makes it possible for the user to *instantly* simulate and understand the behavior of a query over time. While originally designed for extracting historical data, lenses are also appealing in other automated PBD/end-user programming scenarios as they can quickly define many training examples by selecting entire ranges, or individual instances, of valid extractions. As obtaining sufficient training examples is a key problem for automated PDB systems [4], such tools can benefit from historical example generation.

Increasing Robustness

The robustness of end-user programs on the Web is a recognized problem (e.g., [2, 5]) as page templates will likely change at some point [3] causing the program to fail. Because users can interactively detect failures in the extraction, they may refine their selection, adding or removing restrictions, improving filtering conditions or creating another lens for the "failed" interval. The interesting side-effect of the Zoetrope interaction techniques is that although the user may not know how the extraction will fail in the future, they may nonetheless improve the robustness of their extractions by observing failures in the past. Figure 3, for example, depicts the failure of an extraction at some past time point. As the user moves the slider, he finds that in the last state (d) the selection no longer includes the desired data.

Users, as constructors of these programs and extractions, have a unique ability to determine if a program has succeeded. By simulating the program—whether an extraction, or something more sophisticated—on historical data, the programmer can identify failure conditions that may recur in the future and increase the robustness of his programs. User specified failures also give the system an opportunity to automatically adjust its behavior. For example, by identifying a failure on past data, the user is providing examples to the system, which can be used to train a failure detection mechanism. Supplying a "fix" to this failure can help train exception handling mechanisms that will allow a program to continue working or adapting despite future changes to page structure.

Creating a system to store historical Web data, an environment to simulate end-user programs, and creating an appropriate interface to visualize and debug failures will likely require expanding and modifying Zoetrope's behaviors and raises a number of interesting research questions for the future.

Research Questions

Although there are many appealing reasons to support the temporal dimension in Web based end-user programs, there are a number of research issues to address. These include:

Data collection — collecting and storing large sets of temporal data will require new mechanisms for crawling. Although the number of pages that contain useful temporal data might be limited, new systems—possibly a combination of server and P2P—will need to be designed.

Dynamic data and services — Zoetrope is currently targeted at pages that are not dynamically generated (i.e., not the output of form submission). Many end-user programming tasks require submitting input through forms. One might envision that the modified crawling system could contain not just URLs, but also form submission arguments, which may be determined automatically. Nonetheless, when utilizing historical data it is important to consider the implications of mixing live services with historical archives or combining data from pages crawled at different times. For example, converting a list of stock prices from a week ago using the Dollar→Yen conversion tool from today will result in invalid data. While such results are still useful for simulating system behavior, it is important to recognize and help users understand situations in which data is "invalid."

Interfaces and representation — Users are comfortable with the *Now Web*, a familiar and generally unambiguous view of the Web. Introducing time into the interface requires careful consideration of how users might interact with such data. Zoetrope's in-context lens system, while appealing in many ways, allows users to create a view of a page that never existed by setting different lens sliders to different time points. Providing proper affordances and interfaces is crucial for allowing users to properly navigate beyond the *Now*.

Conclusions

The World Wide Web is generally treated by users and system designers as a single snapshot in time. This limited view ignores the importance of temporal data and the potential benefits of maintaining a Web history. By considering the past, end-user programming environments can be enhanced for the future. In this work we have suggested how the components of one temporal system, Zoetrope, can be adapted for use in other situations. In particular, the use of in-context selection lenses allows users to rapidly create examples and enhance the robustness of their programs by labeling examples and fixing errors. The ability to simulate programs on historical data can lead to improved performance but requires additional thought on how such data can be collected, retained, and used.

References

1. Adar, E., Dontcheva, M., Fogarty, J., Weld, D.: Zoetrope: Interacting with the Ephemeral Web. UIST 2008 (2008)

2. Bolin, M., Rha, P., Miller, R.C.: Automation and customization of rendered web pages. UIST 2005 (2005) 163-172

3. Dontcheva, M., Drucker, S.M., Salesin, D., Cohen, M.F.: Changes in Webpage Structure over Time. (2007)

4. Lau, T.: Why PBD systems fail: Lessons learned for usable AI. CHI 2008 Workshop on Usable AI (2008)

5. Leshed, G., Haber, E., Matthews, T., Lau, T.: CoScripter: automating & sharing how-to knowledge in the enterprise. CHI 2008 (2008)